Delimited Persistent Stochastic Non-Interference

Delimited Persistent Stochastic Non-Interference

Jane Hillston
University of Edinburgh, UK
Jane.Hillston@ed.ac.uk

Andrea Marin
Univiversità Ca' Foscari Venezia, Italy
marin@unive.it

Carla Piazza
Università di Udine, Italy
carla.piazza@uniud.it

Sabina Rossi
Università Ca' Foscari Venezia, Italy
sabina.rossi@unive.it

ABSTRACT

Non-Interference is an information flow security property which aims to protect confidential data by ensuring the complete absence of any information flow from high level entities to low level ones. However, this requirement is too demanding when dealing with real applications: indeed, no real policy ever guarantees a total absence of information flow. In order to deal with real applications, it is often necessary to allow mechanisms for downgrading or declassifying information such as information filters and channel control.

In this paper we generalize the notion of *Persistent Stochastic Non-Interference* (*PSNI*) in order to allow information to flow from a higher to a lower security level through a downgrader. We introduce the notion of *Delimited Persistent Stochastic Non-Interference* (*D_PSNI*) and provide two characterizations of it, one expressed in terms of bisimulation-like equivalence checks and another one formulated through unwinding conditions. Then we prove some compositionality properties. Finally, we present a decision algorithm and discuss its complexity.

CCS CONCEPTS

• **Security and privacy → Formal security models**; • **Theory of computation → Algebraic language theory**;

KEYWORDS

Process Algebra, Markovian models, Non-Interference

ACM Reference Format:
Jane Hillston, Andrea Marin, Carla Piazza, and Sabina Rossi. 2019. Delimited Persistent Stochastic Non-Interference. In *12th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2019), March 12–15, 2019, Palma, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3306309.3306329

1 INTRODUCTION

Non-Interference is an information flow security property which aims to protect confidential data by ensuring the complete absence of any information flow from high level entities to low level ones. The concept of non-interference has been introduced by Goguen and Meseguer in [10, 11]. Since then, a large body of work has led to a variety of definitions for different application contexts. A systematic overview is given in [12].

In this paper we consider the notion of non-interference presented in [15] for stochastic, cooperating, processes expressed as terms of the Performance Evaluation Process Algebra (PEPA) [13]. The property *Persistent Stochastic Non-Interference* (*PSNI*) introduced in [15] is based on an observation equivalence that relies on the concept of lumpability ensuring that, for a secure process P, the steady state probability of observing the system being in a specific state P' is independent from its possible high level interactions. In other words, we assume that the observer is able to observe any execution path with its delays and also to measure some timing properties like, e.g., the response time or the throughput. Hence a system S is secure if any external observer is not able to distinguish the behaviour of S performing confidential, high level, activities from the behaviour of the same system but prevented from performing any high level action.

However, as extensively discussed in the literature, absolute non-interference can hardly be achieved in real systems. In order to deal with real applications, it is often necessary to allow mechanisms for downgrading or declassifying information. The aim of our work is that of generalizing the non-interference property *PSNI* in order to admit mechanisms for downgrading or declassifying information such as information filters and channel control. We are not aware of any work dealing with forms of delimited information release in the context of stochastic processes.

Related work. The problem of modelling information flow policies admitting some forms of downgrading has first been addressed by Goguen and Meseguer in [11]. The authors introduce the notion of *conditional non-interference* which admits flows from a high level of security to a low one through a controlled or trusted part. The notion of *intransitive non-interference* has been introduced by Rushby in [23] to formally develop a theory of downgrading for deterministic systems. Indeed, the non-interference property is said to be *intransitive* since flows from the high level to a trusted part and flows from the trusted part to the low level are admissible assuming that the trusted part takes care of controlling them, while a direct flow from high to low is not allowed. In the context of process algebras, intransitive non-interference has been formulated by Roscoe and Goldsmith in [22] for deterministic *CSP* processes.

Intransitive flow policies for non-deterministic systems have been studied by Mantel in [19]. A few years later Backes and Pfitzmann propose in [2] a definition of intransitive *probabilistic* non-interference for reactive systems. In [20] Mullins presents a property

VALUETOOLS 2019, March 12–15, 2019, Palma, Spain

ACM ISBN 978-1-4503-6596-3/19/03…$15.00
https://doi.org/10.1145/3306309.3306329

named *Admissible Interference* for processes expressed as terms of the CCS process algebra.

All the properties mentioned above are based on trace equivalences and thus they are not suitable to deal with information flows caused by possible deadlocks occurring in concurrent computations. To cope with this problem, Ryan and Schneider in [24] introduce the notions of *partial* and *conditional* information flows in the context of CSP processes. Finally, Lafrance and Mullins in [17] introduce the notion of *Bisimulation-based Non-deterministic Admissible Interference* (*BNAI*) which is a generalization of the property presented in [4, 5, 7–9, 21]. In [26] von Oheimb provides automata based definitions for both deterministic and non-deterministic systems and both transitive and intransitive policies. Downgrading in the context of CCS processes has been modelled by some of the authors of this paper in [6].

Contribution of the paper. We present a generalization of the notion of *Persistent Stochastic Non-Interference* (*PSNI*) in order to allow information to flow from a higher to a lower security level through a downgrader. We introduce the notion of *Delimited Persistent Stochastic Non-Interference* (*D_PSNI*) and provide two characterizations of it, one expressed in terms of bisimulation-like equivalence checks and another one formulated through unwinding conditions [3, 18]. As for *PSNI*, the property that we propose is strictly related to the lumping of Markov chains since the observation equivalence at the base of our definition relies on the notion of lumpability [14]. As a consequence if P is secure then, from the low level point of view, the steady state probability of observing the system being in a specific state P' is independent from the possible high level, confidential, interactions of P. Then we prove some compositionality properties and show the relationships between the notions of *PSNI* and *D_PSNI*. Finally, we present a decision algorithm and discuss its complexity.

Structure of the paper. The paper is organized as follows. In Section 2 we introduce the process algebra PEPA and the observation equivalence named *lumpable bisimilarity*. Property *PSNI* and its characterizations are presented in Section 3. Section 4 introduces our novel security property *D_PSNI* and gives two characterizations of it. In Section 5 we describe an algorithm to decide whether a PEPA component is *D_PSNI*. Finally, Section 6 concludes the paper.

2 THE LANGUAGE

In this section we briefly recall the *Performance Evaluation Process Algebra* (PEPA) [13].

Syntax. The PEPA language [13] consists of two basic elements: *components* and *activities*. Activities are pairs (α, r) where α is called *action type* and belongs to a countable set $\mathcal{A}$, while r is called *activity rate* and belongs to the set $R^+ \cup \{\top\}$ where the symbol $\top$ is used to denote an *unspecified* rate. Hence, the duration of an activity is modelled as a negative exponential distribution with mean r^{-1}. The special action type $\tau \in \mathcal{A}$ is used to denote the *unknown* type.

The PEPA language provides a small set of combinators. These allow language terms to be constructed defining the behaviour of components, via the activities they undertake and the interactions between them. The syntax for PEPA terms is given by the following grammar:

$$\begin{aligned} P &::= P \underset{L}{\bowtie} P \mid P/L \mid S \\ S &::= (\alpha, r).S \mid S + S \mid A \end{aligned}$$

where S denotes a *sequential component*, while P denotes a *model component* which can be obtained as the cooperation of sequential terms. We denote by $\mathcal{C}$ the set of all possible components.

Operational semantics. Table 1 shows the operational semantics of the PEPA language. The component $(\alpha, r).P$ carries out the activity (α, r) of type α at rate r and subsequently behaves as component P. $P + Q$ specifies a system which may behave either as P or as Q. $P + Q$ enables all the current activities of both P and Q. The first activity to complete distinguishes one of the components, P or Q. The other component of the choice is discarded. The component P/L behaves as P except that any activity of type within the set L are *hidden*, i.e., they are relabelled with the unknown type τ. The meaning of a constant A is given by a defining equation such as $A \stackrel{def}{=} P$ which gives the constant A the behaviour of the component P. The cooperation combinator $\underset{L}{\bowtie}$ is in fact an indexed family of combinators, one for each possible set of action types, $L \subseteq \mathcal{A} \setminus \{\tau\}$. The *cooperation set* L defines the action types on which the components must synchronise or *cooperate* (the unknown action type, τ, may not appear in any cooperation set). It is assumed that each component proceeds independently with the activities whose types do not occur in the cooperation set L (*individual activities*). However, activities with action types in L require the simultaneous involvement of both components (*shared activities*). The shared activity will have the same action type as the two contributing activities and its rate is that of the slower component. If in a component an activity has rate $\top$, then we say that it is passive with respect to that action type. In this case the rate of the shared activity will be that of the other component. For a given component P and action type α, the *apparent rate* of α in P, $r_\alpha(P)$, is the sum of the rates of the α activities enabled in P.

The semantics of each term in PEPA is given via a labelled *multi-transition system* where the multiplicities of arcs are significant. In the transition system, a state or *derivative* corresponds to each syntactic term of the language and an arc represents the activity which causes one derivative to evolve into another. The set of reachable states of a model P is termed the *derivative set* of P ($ds(P)$) and constitutes the set of nodes of the *derivation graph* of P ($\mathcal{D}(P)$) obtained by applying the semantic rules exhaustively. We denote by $\mathcal{A}(P)$ the set of all the *current action types* of P, i.e., the set of action types which the component P may next engage in. We denote by $\mathcal{A}ct(P)$ the multiset of all the *current activities* of P. Thanks to the exponential assumption, the probability that a particular activity completes is the ratio between its rate and the exit rate from P.

Underlying Markov Chain. Let $P \stackrel{def}{=} P_0$ with $ds(P) = \{P_0, \ldots, P_n\}$ be a finite PEPA model. Then, the stochastic process $X(t)$ on the space $ds(P)$ is a continuous time Markov chain [13]. The *transition rate* between two states P_i and P_j is denoted by $q(P_i, P_j)$ and corresponds to the rate at which the system changes from behaving as component P_i to behaving as P_j, i.e., it is the sum of the activity rates labelling arcs which connect the node corresponding to P_i to the node corresponding to P_j in the derivation graph, i.e.,

$$q(P_i, P_j) = \sum_{a \in \mathcal{A}ct(P_i|P_j)} r_a$$

$$\frac{}{(\alpha, r).P \xrightarrow{(\alpha,r)} P} \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P + Q \xrightarrow{(\alpha,r)} P'} \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P + Q \xrightarrow{(\alpha,r)} Q'}$$

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\alpha,r)} P'/L}\ (\alpha \notin L) \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\tau,r)} P'/L}\ (\alpha \in L) \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{A \xrightarrow{(\alpha,r)} P'}\ (A \stackrel{def}{=} P)$$

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P' \underset{L}{\bowtie} Q}\ (\alpha \notin L) \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P \underset{L}{\bowtie} Q'}\ (\alpha \notin L)$$

$$\frac{P \xrightarrow{(\alpha,r_1)} P' \quad Q \xrightarrow{(\alpha,r_2)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,R)} P' \underset{L}{\bowtie} Q'} \quad R = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q))\ (\alpha \in L)$$

Table 1: Operational semantics for PEPA components

with $P_i \neq P_j$ and $\mathcal{A}ct(P_i|P_j) = \{\!|\, a \in \mathcal{A}ct(P_i) |\ P_i \xrightarrow{a} P_j \,|\!\}$. When P_j is not a one-step derivative of P_i we set $q(P_i, P_j) = 0$. In the following, when possible, we will write q_{ij} instead of $q(P_i, P_j)$. In the definition of the infinitesimal generator $\mathbf{Q}$ of $X(t)$, the q_{ij}'s, with $i \neq j$, are the off-diagonal elements of the matrix whereas the diagonal elements are the negative sum of the row non-diagonal elements, i.e., $q_{ii} = -q(P_i)$. For any finite and irreducible PEPA model P, the steady-state distribution $\Pi(\cdot)$ exists and it may be found by solving the probability normalising equation and the linear system of global balance equations:

$$\textstyle\sum_{P_i \in ds(P)} \Pi(P_i) = 1 \text{ and } \Pi\mathbf{Q} = \mathbf{0}.$$

Another notion that will be used in the paper is that of *conditional transition rate* from P_i to P_j via an action type α, denoted by $q(P_i, P_j, \alpha)$. This is the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph which are also labelled by the action type α. It is the rate at which a system behaving as component P_i evolves to behaving as component P_j as the result of completing a type α activity. The *total conditional transition rate* from P to $S \subseteq ds(P)$, $q[P, S, \alpha]$, is defined as

$$q[P, S, \alpha] = \sum_{P' \in S} q(P, P', \alpha)$$

where $q(P, P', \alpha) = \sum_{P \xrightarrow{(\alpha, r_\alpha)} P'} r_\alpha$.

Observation Equivalence. We consider a bisimulation-like equivalence notion for PEPA components, named *lumpable bisimilarity*, that we previously introduced in [14].

Two PEPA components are *lumpably bisimilar* if there exists an equivalence relation between them such that, for any action type α different from τ, the total conditional transition rates from those components to any equivalence class, via activities of this type, are the same.

Definition 2.1. (Lumpable bisimulation) An equivalence relation over PEPA components, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, is a *lumpable bisimulation* if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$ such that

- either $\alpha \neq \tau$,
- or $\alpha = \tau$ and $P, Q \notin S$,

it holds

$$q[P, S, \alpha] = q[Q, S, \alpha]\,.$$

Notice that, in contrast with the notion of strong equivalence [13], lumpable bisimulation allows arbitrary activities with type τ among components belonging to the same equivalence class, and therefore it is less strict.

We are interested in the relation which is the largest lumpable bisimulation, that is the union of all lumpable bisimulations.

Definition 2.2. (Lumpable bisimilarity) Two PEPA components P and Q are *lumpably bisimilar*, written $P \approx_l Q$, if $(P, Q) \in \mathcal{R}$ for some lumpable bisimulation $\mathcal{R}$, i.e.,

$$\approx_l = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a lumpable bisimulation}\}.$$

$\approx_l$ is called *lumpable bisimilarity* and it is the largest symmetric lumpable bisimulation over PEPA components.

We proved that for any PEPA component P, lumpable bisimilarity induces a partition of the derivative set $ds(P)$ of P into equivalence classes that is a *strong lumpability* [16] for the underlying Markov chain. Moreover, the aggregated process satisfies the property that the steady state probability of each aggregated macro-state is equal to the sum of the steady state probabilities of the corresponding equivalent states in the initial CTMC.

Finally in [14] we proved that lumpable bisimilarity is a congruence, i.e., if $P_1 \approx_l P_2$ then

- $(\alpha, r).P_1 \approx_l (\alpha, r).P_2$ for all $\alpha \in \mathcal{A}$;
- $P_1/L \approx_l P_2/L$ for all $L \subseteq \mathcal{A}$;
- $P_1 \underset{L}{\bowtie} Q \approx_l P_2 \underset{L}{\bowtie} Q$ for all $L \subseteq \mathcal{A}$.

3 STOCHASTIC NON-INTERFERENCE

In this section we recall the security property named *Persistent Stochastic Non-Interference (PSNI)* for PEPA components which aim

at characterizing classes of processes having no information flows from high to low.

Property *PSNI* tries to capture every possible information flow from a *classified (high)* level of confidentiality to an *untrusted (low)* one. The definition of *PSNI* is based on the basic idea of Non-Interference [10]: "No information flow is possible from high to low if what is done at the high level *cannot interfere* in any way with the low level". Hence, the notion of *PSNI* consists of checking all the states reachable by the system against all high level potential interactions.

In order to formally define this security property, we partition the set $\mathcal{A} \setminus \{\tau\}$ of visible action types, into two sets, $\mathcal{H}$ and $\mathcal{L}$, of high and low level action types, respectively. A high level PEPA component H is a PEPA term such that for all $H' \in ds(H)$, $\mathcal{A}(H') \subseteq \mathcal{H}$, i.e., every derivative of H may next engage in only high level actions. We denote by $\mathcal{C}_H$ the set of all high level PEPA components.

A system P satisfies property *PSNI* if for every state P' reachable from P and for every high level process H a low level user cannot distinguish P' running in isolation or, equivalently, P' running in parallel with any high level PEPA component that does not synchronize with it from $P' \underset{\mathcal{H}}{\bowtie} H$ where H is a high component cooperating with P'. In other words, a system P satisfies *PSNI* if what a low level user sees of the system is not modified when it cooperates with any high level process H.

In order to model a process P running in isolation we introduce the special model component $\mathbf{0}$ and extend the syntax of model, non-sequential, PEPA components as

$$P \quad ::= \quad \mathbf{0} \mid P \underset{L}{\bowtie} P \mid P/L \mid S\,.$$

We model a process P running in isolation or, equivalently, running in parallel with any high level PEPA component that does not synchronize with it as $(P \underset{\mathcal{H}}{\bowtie} \mathbf{0})$. Now observe that high level interactions are not visible by an external low level observer which is only aware of the delay while the interactions take place. Therefore, from the low level point of view the process $(P \underset{\mathcal{H}}{\bowtie} H)$ behaves as $(P \underset{\mathcal{H}}{\bowtie} H)/\mathcal{H}$.

We are now in position to formally define the property *PSNI*. The observation equivalence at the base of our definition relies on the notion of lumpable bisimilarity and this ensures that, for a secure process P, the steady state probability of observing the system being in a specific state P' is independent from its possible high level interactions.

Definition 3.1. Let P be a PEPA component.

$$P \in PSNI \text{ iff } \forall P' \in ds(P),\ \forall H \in \mathcal{C}_H,$$

$$(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})/\mathcal{H} \approx_l (P' \underset{\mathcal{H}}{\bowtie} H)/\mathcal{H}\,.$$

Notice that $(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})$ does not engage in any high level activity and then the derivation graphs of $(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})$ and $(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})/\mathcal{H}$ are isomorphic as graphs with labels and edges. More precisely, $(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})$ behaves as the component P' prevented from performing any high level activity. We can thus express the behaviour of $(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})$ by using the restriction operator $(\setminus)$ as defined for instance in CCS, as $P \setminus \mathcal{H}$. From now on we use the notation $P \setminus \mathcal{H}$ as a shorthand of $(P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})/\mathcal{H}$. The definition of *PSNI* can be expressed as:

Definition 3.2. [15] Let P be a PEPA component.

$$P \in PSNI \text{ iff } \forall P' \in ds(P),\ \forall H \in \mathcal{C}_H,$$

$$P' \setminus \mathcal{H} \approx_l (P' \underset{\mathcal{H}}{\bowtie} H)/\mathcal{H}\,.$$

In [15] we provided two characterizations of *PSNI*: one based on bisimulation-like equivalence checks and another one based on unwinding conditions which demand properties of individual actions.

The first characterization of *PSNI* is based on an observation equivalence where actions from $\mathcal{H}$ *may* be ignored. We introduced the notion of *lumpable bisimilarity up to* $\mathcal{H}$.

Definition 3.3. (Lumpable bisimilarity up to $\mathcal{H}$) An equivalence relation over PEPA components, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, is a *lumpable bisimulation up to* $\mathcal{H}$ if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$

- if $\alpha \notin \mathcal{H} \cup \{\tau\}$ then
$$q[P, S, \alpha] = q[Q, S, \alpha]\,,$$
- if $\alpha \in \mathcal{H} \cup \{\tau\}$ and $P, Q \notin S$, then
$$q[P, S, \alpha] = q[Q, S, \alpha]\,.$$

Two PEPA components P and Q are *lumpably bisimilar up to* $\mathcal{H}$, written $P \approx_l^{\mathcal{H}} Q$, if $(P, Q) \in \mathcal{R}$ for some lumpable bisimulation up to $\mathcal{H}$, i.e.,

$$\approx_l^{\mathcal{H}} = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a lumpable bisimulation up to } \mathcal{H}\}.$$

$\approx_l^{\mathcal{H}}$ is called *lumpable bisimilarity up to* $\mathcal{H}$ and it is the largest symmetric lumpable bisimulation up to $\mathcal{H}$ over PEPA.

The first characterization of *PSNI* is stated below.

Theorem 3.4. *Let P be a PEPA component. Then*

$$P \in PSNI \text{ iff } P \setminus \mathcal{H} \approx_l^{\mathcal{H}} P\,.$$

We also provided a characterization of *PSNI* in terms of *unwinding conditions*. In practice, whenever a state P' of a *PSNI* PEPA model P may execute a high level activity leading it to a state P'', then P' and P'' are indistinguishable for a low level observer.

Theorem 3.5. *Let P be a PEPA component.*

$$P \in PSNI \text{ iff} \forall P' \in ds(P),$$

$$P' \xrightarrow{(h,r)} P'' \text{ implies } P' \setminus \mathcal{H} \approx_l P'' \setminus \mathcal{H}\,.$$

Using the equivalence relation $\approx_l^{\mathcal{H}}$ this can be reformulated as follows.

Theorem 3.6. *Let P be a PEPA component.*

$$P \in PSNI \text{ iff} \forall P' \in ds(P),$$

$$P' \xrightarrow{(h,r)} P'' \text{ implies } P' \approx_l^{\mathcal{H}} P''\,.$$

Example 3.7. The aim of this first example is to support the intuition underlying the definition of PSNI. Let us consider a system offering some public access to two types of customers: the first consists of ordinary customers without any special privilege, while the second is formed by the users with administrative privileges who can access to some confidential information stored in the system. A malicious user is not interested in attacking the system

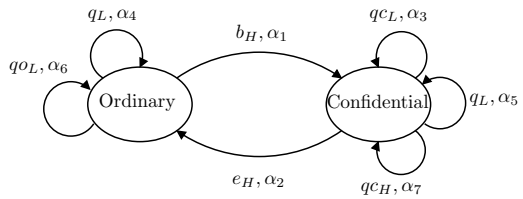

Figure 1: An unsecure system for public access.

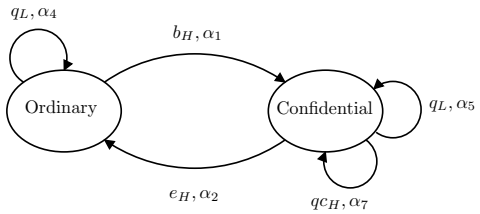

Figure 2: A secure system for public access.

while ordinary users are interacting since a successful attempt would lead to an irrelevant leakage of information. On the contrary, an attack while the system is in its 'Confidential' state may lead to a valuable leakage of information. Hence, it becomes crucial for the system to hide its operating state to low-level users at any time epoch.

The model is depicted in Figure 1. High-level users can force the system to move from the state 'Ordinary' to the state 'Confidential' and vice-versa with the actions b_H and e_H (standing for 'begin' and 'end', respectively). Once the system is in the 'Confidential' state, the high-level user interacts by means of the action type qc_H (standing for query at confidential state). A low-level user can interact with the system in the following ways:

- Action types qc_L and qo_L (query at confidential and at ordinary, respectively) functionally allow the user to understand that the system is in the 'Confidential' or 'Ordinary' state: for example, a request for service could produce a message displaying that an update is being run if the system is in the 'Confidential' state or the expected reply otherwise.
- Action type q_L allows the low-level user to interact with the system without being able to distinguish its functionality from that observed when the system state is 'Ordinary'. In fact, the action type q_L is exposed in both state 'Ordinary' and 'Confidential'.

It is clear that the system, in this form, is not secure. In fact, action types qc_L and qo_L simply allow the malicious user to understand its state. Indeed, the states 'Ordinary' and 'Confidential' are not lumpable bisimilar for a low level observer since they expose a different set of low-level actions.

Let us consider the model depicted in Figure 2 that is obtained from the one of Figure 1 by suppressing the action types qc_L and qo_L. The obtained model satisfies PSNI iff $\alpha_4 = \alpha_5$. If this is not the case, the malicious attacker may infer the state of the system thanks to the sequence of response times obtained by a set of queries. In other words, the state of the system can be inferred not only by its functional properties but also by its non-functional properties.

4 DELIMITED NON-INTERFERENCE

The notion of *PSNI* is too demanding when dealing with practical applications: indeed no real policy ever guarantees a total absence of information flow. In many concrete applications confidential data can flow from high to low provided that the flow is not direct and it is controlled by the system, i.e., a trusted part of the system can control the downgrading of sensitive information.

In this section we show how our security property can be generalized in order to obtain a notion of stochastic non-interference for PEPA components which allows systems to intentionally release some information.

To model downgrading we now partition the set $\mathcal{A} \setminus \{\tau\}$ of visible action types, into three sets, $\mathcal{H}$, $\mathcal{L}$ and $\mathcal{D}$ of high, low and downgraded action types. Downgraded action types are used to specify the behaviour of trusted components interacting with the system. We assume that the low level users cannot observe the actions performed by the trusted part.

We generalize the notation expressed in terms of $(\setminus)$ in the previous section. Indeed, for a given PEPA component P and a set of action types $L \in \mathcal{A} \setminus \{\tau\}$, we denote by $P \setminus L$ the component P prevented from performing any activity whose action type belongs to L.

Delimited Persistent Stochastic Non-Interference (*D_PSNI*) can be formalized as follows.

Definition 4.1. Let P be a PEPA component.

$$P \in D\_PSNI \text{ iff } \forall P' \in ds(P),\ \forall H \in \mathcal{C}_H,$$
$$((P' \underset{\mathcal{H}}{\bowtie} \mathbf{0})/\mathcal{H}) \setminus \mathcal{D} \approx_l ((P' \underset{\mathcal{H}}{\bowtie} H)/\mathcal{H}) \setminus \mathcal{D}\,.$$

Notice that this definition states that a system P satisfies *D_PSNI* if *whenever it does not cooperate with a trusted part*, what a low level user sees of the system is not modified when it cooperates with any high level process H. Hence, flows from the high level to the trusted part and flows from the trusted part to the low level are admissible, while direct flows from the high level to the low one are not allowed.

As in the case of *PSNI*, property *D_PSNI* can be equivalently written as:

Definition 4.2. Let P be a PEPA component.

$$P \in D\_PSNI \text{ iff } \forall P' \in ds(P),\ \forall H \in \mathcal{C}_H,$$
$$P' \setminus \mathcal{H} \cup \mathcal{D} \approx_l ((P' \underset{\mathcal{H}}{\bowtie} H)/\mathcal{H}) \setminus \mathcal{D}\,.$$

In this section we provide two characterizations of *D_PSNI*. The first one is expressed in terms of a bisimulation-like equivalence relation named $\mathcal{L}$*-Lumpable bisimilarity up to* $\mathcal{H}$.

Definition 4.3. ($\mathcal{L}$-Lumpable bisimilarity up to $\mathcal{H}$) An equivalence relation over PEPA components, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, is a $\mathcal{L}$*-lumpable bisimulation up to* $\mathcal{H}$ if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{L} \cup \mathcal{H} \cup \{\tau\}$ and for all $S \in \mathcal{C}/\mathcal{R}$

- if $\alpha \in \mathcal{L}$ then
$$q[P, S, \alpha] = q[Q, S, \alpha]\,,$$
- if $\alpha \in \mathcal{H} \cup \{\tau\}$ and $P, Q \notin S$, then
$$q[P, S, \alpha] = q[Q, S, \alpha]\,.$$

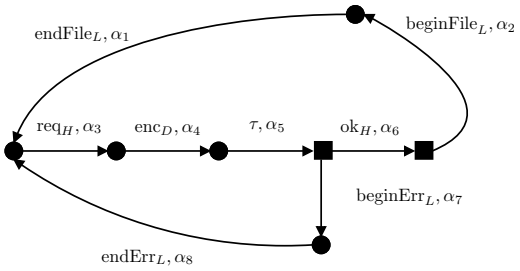

Figure 3: Unsecure database query.

Two PEPA components P and Q are *$\mathcal{L}$-lumpably bisimilar up to $\mathcal{H}$*, written $P \approx^{\mathcal{H}}_{\mathcal{L}} Q$, if $(P, Q) \in \mathcal{R}$ for some $\mathcal{L}$-lumpable bisimulation up to $\mathcal{H}$, i.e.,

$$\approx^{\mathcal{H}}_{\mathcal{L}} = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a } \mathcal{L}\text{-lumpable bisimulation up to } \mathcal{H}\}.$$

$\approx^{\mathcal{H}}_{\mathcal{L}}$ is called *$\mathcal{L}$-lumpable bisimilarity up to $\mathcal{H}$* and it is the largest symmetric $\mathcal{L}$-lumpable bisimulation up to $\mathcal{H}$ over PEPA components.

The next theorem states that P satisfies *D_PSNI* if and only if for all $P' \in ds(P)$ it holds that P' and $P' \setminus \mathcal{H} \cup \mathcal{D}$ are indistinguishable with respect to $\approx^{\mathcal{H}}_{\mathcal{L}}$.

Theorem 4.4. *Let P be a PEPA component. Then $P \in$ D_PSNI iff $\forall P' \in ds(P)$*

$$P' \setminus \mathcal{H} \cup \mathcal{D} \approx^{\mathcal{H}}_{\mathcal{L}} P'.$$

We also provide a characterization of *D_PSNI* in terms of *unwinding conditions* which demand properties of individual activities. This characterization of *D_PSNI* is stated below.

Theorem 4.5. *Let P be a PEPA component.*

$$P \in \text{D\_PSNI iff} \forall P' \in ds(P),$$

$$P' \xrightarrow{(h,r)} P'' \text{ implies } P' \setminus \mathcal{H} \cup \mathcal{D} \approx_l P'' \setminus \mathcal{H} \cup \mathcal{D}.$$

Theorems 4.4 and 4.5 provide different characterizations of *D_PSNI* which naturally lead to efficient methods for the verification and construction of secure systems. A decision algorithm for *D_PSNI* is presented in Section 5.

Example 4.6. Let us consider the model depicted in Figure 3 that represents a confidential query to a database system. Starting from the left-most state, the system waits for a request req$_H$ that uses a private channel, e.g., a channel based on a asymmetric cryptography. Upon the reception of the request, the system negotiates a symmetric key that will be used to transfer the reply (enc$_D$). This phase is observable by a malicious user, but by using the downgrading we are stating that we tolerate the information flow that happens up to this point. The following τ activity represents the computation of the reply. At this point, we may observe two behaviours: either the computation is successful or it is unsuccessful. In the former case, the system transmits on the private channel the acknowledge ok$_H$, then begins (beginFile$_L$) and ends (endFile$_L$) the transmission of the reply encrypted with the shared key negotiated before. For this reason, these activities are modelled by means of low-level action types. In the case of failure, the system transmits an error message (beginErr$_L$, endErr$_L$).

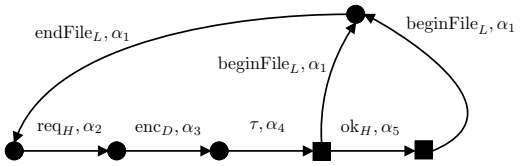

Figure 4: Secure database query.

Let us analyse the information flow in the system. Notice that once a malicious observer sees the action type enc$_D$, he/she can infer that a query has been started. However, thanks to the downgrading, we consider that this flow is acceptable because we trust the security of the system in this phase. In contrast, action type ok$_H$ is not protected (although cannot be seen by the malicious user) since it is followed by a beginFile$_L$ action type. Therefore, the observer can deduce the state of the query by inferring that it has been successful if it sees beginFile$_L$ after enc$_D$, or unsuccessful if it sees beginErr$_L$ after enc$_D$. Formally, this can be seen since the states depicted by squares in Figure 3 are not lumpable bisilimilar from a low-level point of view.

One possible solution to this leakage of information is shown in Figure 4. First of all, we avoid the use of different action types for signalling the error. This means, for example, that the error message must use the same service (e.g., TCP port) as the correct reply message. Second, it is important that the distribution of the size of the reply and error message is the same (in our case proportional to an exponential random variable with parameter α_1). If this is not the case, a malicious observer could be able to probabilistically infer the reply the outcome of the query (successful, unsuccessful) by the transmission time of the reply.

Here we prove some compositionality results that allow us to check the security of a system by only verifying the security of its subcomponents. In particular we prove that *D_PSNI* is compositional with respect to non-high prefix, hiding, and cooperation over a set of low actions.

Proposition 4.7. *Let P and Q be two PEPA components. If $P, Q \in$ D_PSNI, then*

- $(\alpha, r).P \in$ *D_PSNI for all* $\alpha \in \mathcal{L} \cup \mathcal{D} \cup \{\tau\}$;
- $P/L \in$ *D_PSNI for all* $L \subseteq \mathcal{A}$;
- $P \underset{L}{\bowtie} Q \in$ *D_PSNI for all* $L \subseteq \mathcal{L}$.

We also prove that if $P \in$ *D_PSNI* then the equivalence class $[P]$ with respect to lumpable bisimilarity $\approx_l$ is closed under *D_PSNI*.

Proposition 4.8. *Let P and Q be two PEPA components. If $P \in$ D_PSNI and $P \approx_l Q$ then also $Q \in$ D_PSNI.*

We conclude this section by showing the relationships between the two equivalence relations $\approx^{\mathcal{H}}_{l}$ and $\approx^{\mathcal{H}}_{\mathcal{L}}$ and also between the two security properties *PSNI* and *D_PSNI*.

Proposition 4.9 relates the bisimulation-like equavalence relations $\approx^{\mathcal{H}}_{l}$ and $\approx^{\mathcal{H}}_{\mathcal{L}}$. The proof follows immediately from Definitions 3.3 and 4.3.

Proposition 4.9. *Let P and Q be two PEPA components. It holds that*

$$P \approx^{\mathcal{H}}_{\mathcal{L}} Q \text{ iff } P \setminus \mathcal{D} \approx^{\mathcal{H}}_{l} Q \setminus \mathcal{D}.$$

The next Proposition gives a characterization of *D_PSNI* in terms of $\approx^{\mathcal{H}}_{l}$. The proof is straightforward and follows from Proposition 4.9.

Proposition 4.10. *Let P be a PEPA component. Then $P \in$ D_PSNI iff $\forall P' \in ds(P)$ it holds that*

$$P' \setminus \mathcal{H} \cup \mathcal{D} \approx^{\mathcal{H}}_{l} P' \setminus \mathcal{D}.$$

Finally, we show how *D_PSNI* can be expressed in terms of *PSNI*.

Proposition 4.11. *Let P be a PEPA component.*

$$P \in \text{D\_PSNI iff } \forall P' \in ds(P),\ P' \setminus \mathcal{D} \in \text{PSNI}.$$

5 A DECISION ALGORITHM

We briefly describe an algorithm to decide whether a PEPA component having a finite set of derivatives is *D_PSNI*.

In [1] an algorithm has been presented for solving the label-compatibility problem. The algorithm works on directed labelled weighted graphs defined as follows.

Definition 5.1. (Directed labelled weighted graph) A directed labelled weighted graph is a tuple $G = (V, Lab, E, w)$ where:

- V is a finite set of vertices;
- Lab is a finite set of labels;
- $E \subseteq V \times V \times Lab$ is a finite set of labelled edges;
- $w : E \to \mathbb{R}$ is a weighting function that associates a value to each edge.

Given $V' \subseteq V$, we denote by $w(v, V', a)$ the sum of the weights of the edges from v to V' having label a.

The label-compatibility problem is an extension to labelled weighed graphs of the problem described in [25].

Definition 5.2. (Label-Compatibility Problem) Let $G = (V, Lab, E, w)$ be a directed labelled weighted graph and $\mathcal{R} \subseteq V \times V$ be an equivalence relation over V. $\mathcal{R}$ is said to be *label-compatible* with G if for each $a \in Lab$, for each $C, C' \in V/\mathcal{R}$, and for each $v, v' \in C$ it holds that $w(v, C', a) = w(v', C', a)$. Moreover, the *labelled weighted compatibility problem over G* requires to compute the largest equivalence relation label-compatible with G.

In [1] it has been presented an algorithm, named $LCW(\_)$, for solving the label-compatibility problem on G and it has been proved that the label-compatibility problem has always a unique solution which can be computed in time $O(|V|+|E| \log |V|)$. Here we consider the algorithm $LCW(\_)$ described in [1] where the initial relation is the total relation.

The problem of deciding $P \approx_l Q$ can be reduced to a label-compabitility problem. Hence, by Theorem 4.5 we immediately get an algorithm for deciding $P \in$ *D_PSNI* in polynomial time with respect to the dimension of $ds(P)$. This algorithm consists of evaluating whether $P' \setminus \mathcal{H} \cup \mathcal{D} \approx_l P'' \setminus \mathcal{H} \cup \mathcal{D}$, for each $P', P'' \in ds(P)$ such that $P' \xrightarrow{(h,r)} P''$.

However, we can improve on the above algorithm and reduce the problem of deciding $P \in$ *D_PSNI* to a single run of $LCW(\_)$ followed by a final check. In particular, for a given PEPA component P let us consider the following graph.

Algorithm 1 Algorithm for D_PSNI

```
1:  function DPSNI(D(P))
2:      Compute Down(P)
3:      P = LCW(Down(P))
4:      for P' --(h,r)--> P'' in D(P) do
5:          if (P', P'') ∉ P then
6:              return False
7:          end if
8:      end for
9:      return True
10: end function
```

Definition 5.3. (Downgrading Graph) Let P be PEPA component. The *downgrading graph of P* is the directed labelled weighted graph $Down_P = (VD_P, \mathcal{L} \cup \{\tau\}, ED_P, wd_P)$, where:

- $VD_P = \{R \mid R \text{ is } P' \setminus \mathcal{H} \cup \mathcal{D} \text{ with } P' \in ds(P)\}$
- ED_P is the set of labelled edges

$$ED_P = \begin{array}{l} \{(R, R', \alpha) \mid R \xrightarrow{(\alpha,r)} R' \text{ with } \alpha \in \mathcal{L} \cup \{\tau\}\} \\ \cup \{(R, R, \tau) \mid R \in VD_P\}\} \end{array}$$

- wd_P is the function which associates to each edge in ED_P the value

$$wd_P(R, R', \alpha) = \begin{cases} q(R, R', \alpha) \text{ if } \alpha \neq \tau \vee R \neq R' \\ -q[R, VD_P \setminus \{R\}, \alpha] \text{ otherwise} \end{cases}$$

Notice that the downgrading graph is not necessarily connected. The algorithm $LCW(Down_P)$ returns a binary relation $\mathcal{P}$. In order to decide whether $P \in$ *D_PSNI* we have to check that whenever $P' \xrightarrow{(h,r)} P''$ is an edge in the derivation graph $\mathcal{D}(P)$, then $(P', P'') \in \mathcal{P}$. This test is performed by Algorithm 1 that computes $DPSNI(\mathcal{D}(P))$. The computation of $Down(P)$ at line 2 can be performed in linear time with respect to the size of $\mathcal{D}(P)$ by applying the rules of Definition 5.3. Notice that the graph $Down(P)$ has size at most equal to the size of $\mathcal{D}(P)$. Once $LCW(Down(P))$ has computed the relation $\mathcal{P}$ the for-loop at lines 4-8 checks that the conditions of Theorem 4.5 are satisfied.

Let E_P be the transitions in $\mathcal{D}(P)$. The following theorem states the correctness of our algorithm and evaluates its time complexity.

Theorem 5.4 (Correctness and Complexity). *Let P be a PEPA component having a finite set of derivatives. The algorithm $DPSNI(\mathcal{D}(P))$ terminates in time*

$$O(|ds(P)| + |E_P| + |ED_P| \log(|VD_P|))$$

and returns True if and only if P is D_PSNI.

6 CONCLUSION

In this paper we presented a form of *delimited persistent* information flow security property for stochastic processes specified as terms of a quantitative process algebra, namely Performance Evaluation Process Algebra (PEPA). Our property *D_PSNI* is based on a bisimulation based observation equivalence for the PEPA terms which induces a lumping on the underlying Markov chain. The aim of our definition is that of protecting systems from maliciuos

attackers which are able to measure also the timing properties of the system, e.g., the response time or the throughput. Property *D_PSNI* implements a notion of intransitive non-interference: flows from the high level to a trusted part and flows from the trusted part to the low level are admissible since they are intended to be controlled by the trusted part, while a direct flow from high to low is not allowed.

In this paper we also deal with compositionality issues and prove that *D_PSNI* is compositional with respect to non-high prefix, hiding and cooperation over low level actions. The relationships between *PSNI* and *D_PSNI* are formally stated. Moreover, a decision algorithm for *D_PSNI* is presented

As a future work we plan to relax the definition of Non-Interference by introducing metrics that allow us to measure the security degree of a system in terms of probabilities.

ACKNOWLEDGMENTS

The work described in this paper has been partially supported by the Università Ca' Foscari Venezia - DAIS within the IRIDE program, by the Università di Udine PRID ENCASE project, and by GNCS-INdAM.

REFERENCES

[1] G. Alzetta, A. Marin, C. Piazza, and S. Rossi. 2018. Lumping-based equivalences in Markovian automata: Algorithms and applications to product-form analyses. *Information and Computation* 260 (2018), 99–125.

[2] M. Backes and B. Pfitzmann. 2003. Intransitive Non-Interference for Cryptographic Purposes. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'03)*. IEEE, 140–152.

[3] A. Bossi, R. Focardi, D. Macedonio, C. Piazza, and S. Rossi. 2004. Unwinding in Information Flow Security. *Electr. Notes Theor. Comput. Sci.* 99 (2004), 127–154.

[4] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. 2003. Bisimulation and Unwinding for Verifying Possibilistic Security Properties. In *Proc. of Int. Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'03) (LNCS)*, L. D. Zuck, P. C. Attie, A. Cortesi, and S. Mukhopadhyay (Eds.), Vol. 2575. Springer-Verlag, 223–237.

[5] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. 2003. A Proof System for Information Flow Security. In *Logic Based Program Development and Transformation (LNCS)*, M. Leuschel (Ed.), Vol. 2664. Springer-Verlag, 199–218.

[6] A. Bossi, C. Piazza, and S. Rossi. 2004. Modelling Downgrading in Information Flow Security. In *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*. IEEE, 187–201.

[7] A. Bossi, C. Piazza, and S. Rossi. 2007. Compositional Information Flow Security for Concurrent Programs. *Journal of Computer Security* 15, 3 (2007), 373–416.

[8] R. Focardi and S. Rossi. [n. d.]. A Security Property for Processes in Dynamic Contexts (extended abstract). ([n. d.]). In *Proc. of Workshop on Issues in the Theory of Security (WITS'02)*.

[9] R. Focardi and S. Rossi. 2006. Information flow security in dynamic contexts. *Journal of Computer Security* 14, 1 (2006), 65–110.

[10] J. A. Goguen and J. Meseguer. 1982. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'82)*. 11–20.

[11] J. A. Goguen and J. Meseguer. 1984. Unwinding and Inference Control. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'84)*. 75–86.

[12] D. Hedin and A. Sabelfeld. 2012. A Perspective on Information-Flow Control. In *Software Safety and Security - Tools for Analysis and Verification*. IOS Press, 319–347.

[13] J. Hillston. 1996. *A Compositional Approach to Performance Modelling*. Cambridge Press.

[14] J. Hillston, A. Marin, C. Piazza, and S. Rossi. 2013. Contextual Lumpability. In *Proc. of Valuetools 2013 Conf.* ACM Press, 194–203.

[15] J. Hillston, A. Marin, C. Piazza, and S. Rossi. 2018. Information Flow Security for Stochastic Processes. In *Computer Performance Engineering - 15th European Workshop, EPEW*. 142–156.

[16] J. G. Kemeny and J. L. Snell. 1960. *Finite Markov Chains*. D. Van Nostrand Company, Inc.

[17] S. Lafrance and J. Mullins. 2002. Bisimulation-based Non-deterministic Admissible Interference and its Application to the Analysis of Cryptographic Protocols. *Electronic Notes in Theoretical Computer Science* 61 (2002), 1–24.

[18] H. Mantel. 2000. Unwinding Possibilistic Security Properties. In *Proc. of the European Symposium on Research in Computer Security (ESoRiCS'00) (LNCS)*, Vol. 2895. Springer-Verlag, 238–254.

[19] H. Mantel. 2001. Information Flow Control and Applications - Bridging a Gap. In *Proc. of the nternational Symposium of Formal Methods Europe (FME'01) (LNCS)*. Springer-Verlag, 153–172.

[20] J. Mullins. 2000. Nondeterministic Admissible Interference. *Journal of Universal Computer Science* 11 (2000), 1054–1070.

[21] C. Piazza, E. Pivato, and S. Rossi. 2004. CoPS - Checker of Persistent Security. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*. 144–152.

[22] A. W. Roscoe and M. H. Goldsmith. 1999. What Is Intransitive Noninterference?. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'99)*. 228–238.

[23] J. Rushby. 1992. *Noninterference, Transitivity, and Channel-Control Security Policies*. Technical Report CSL-92-02. SRI International.

[24] P.Y.A. Ryan and S. Schneider. 2001. Process Algebra and Non-Interference. *Journal of Computer Security* 9, 1/2 (2001), 75–103.

[25] A. Valmari and G. Franceschinis. 2010. Simple O(m logn) Time Markov Chain Lumping. In *Proc. of Int. Conf. TACAS*, Vol. 6015. Springer Verlag, 38–52.

[26] David von Oheimb. 2004. Information Flow Control Revisited: Noninfluence = Noninterference + Nonleakage. In *Computer Security (ESORICS'04)*. 225–243.